Final Project for Intelligent Robotics

Group 8

Abstract

In the time of the pandemic, social distancing is encouraged for everyone. Keeping this in mind, our knowledge of intelligent robotics helps reduce social contact between people in one way. We're proposing an autonomous robot that helps navigate people to their destination. The key components of our robot are human robot interaction, path planning, robot movement and autonomous navigation. In this paper we cover our methods, goals, experimentations and evaluations and future goals.

Introduction

The Covid-19 pandemic is having a profound effect on our lives and psyche and we tend to avoid any physical contact. For new students coming to university, going around campus can get very difficult as they don't know the different school buildings, lecture halls, specific offices within buildings. With the help of advancement in the field of robotics, we can switch the robot from playing a passive role to an active role in our day-to-day lives for the betterment of us. Taking advantage of increasing knowledge on robotics and using it to avoid close proximity between strangers, we plan to develop an autonomous robot that can help guide an individual through unfamiliar places.

The project aims to develop an autonomous robot that helps an individual to find a destination by walking them through the path. The robot will get the destination from the user through speech recognition and communicate with the user to guide them to the destination. The robot will find the shortest path to the destination and guide the user to the destination. Once it reaches the destination the robot announces that the destination has arrived and after waiting there for a brief period it returns to the base/original position.

One of the reasons we choose to do this project is because we get hands-on experience with state-of-the-art robotics and developing complex systems which work together. Also, along with that, this also helps contribute for betterment to society.

Project Goals

Table below Provides the basic goals of this project with the appropriate steps required to achieve them. Furthermore, the *Outcome* indicated whether or not the goal has successfully been completed.

Goals	How it would be achieved	Outcome
Autonomous navigation	Can be achieved using rospy	✓
Obstacle avoidance: used in order for the robot to avoid any unknown obstacles	With the help of laser scan we detect unknown object on the path and find a new path	X
Human computer interaction	Using Google SpeechRecognition API and Python Text-To-Speech API.	✓
Path planning	Using A*	✓

Related Work

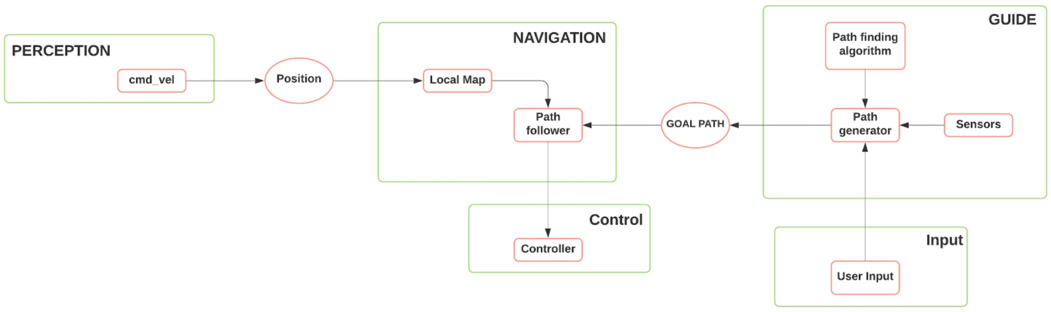
1. In the paper “A tour-guide robot: Moving towards interaction with humans”[1], they developed a smart robot that can work as a tour guide in different environments. This is done with the help of hardware components like a laser, cameras, platform, face and voice detection and so on. An extended Kalman filter is used for localisation. We attempt to create a subset of this idea to cover a smaller area.
2. In the paper “Robotic Guide Dog”[2], they developed a robot dog that helped guide people with visual disabilities to their location. This work proposes one of the first end-to-end human-robot interaction systems to serve as an autonomous navigation aid. Our future goal is to extend this concept in order to develop an assistant for the visually impaired.
3. In the paper “AUTONOMOUS UV ROBOT WITH SLAM”[3] they’ve built an autonomous robot based which implements SLAM that moves around lobbies and hallways to get into rooms to disinfect using UV. The communication with the robot was done using a mobile app. In our future goal to develop a security robot, we’ll use SLAM in order for the robot to explore regions not in the local map.
4. In the paper “SLAM algorithm applied to robotics assistance for navigation in unknown environments”[4], They have improved the autonomy of motor disabled people by implementation of general MCI which could be adapted to users capabilities so the robot could be navigated through unknown environments. The SLAM is based on Extended Kalman Filter. The logic of how we would move the robot was inspired by this project.

Contributions

In this project we have added many features, some of which are done with the help of libraries and others implemented by us. In the framework description we went in depth of how each of them were implemented. In the table below, we put out which of these were implemented by us and which ones are with the help of the library.

Component	Implemented by us (yes/no)
1: Autonomous navigation	Yes
2: Speech recognition	No
3: Text-to-speech	No
4: Path planning	Yes

Framework description



The above component diagram details how each of the components work with each other.

1. Path Planning and Autonomous Navigation

A significant component and contribution in our project is primarily used for the robot in determining an optimal path from initial point in the map to the destination using the *A * algorithm*. The path generation strategy involves mathematics and data structures from

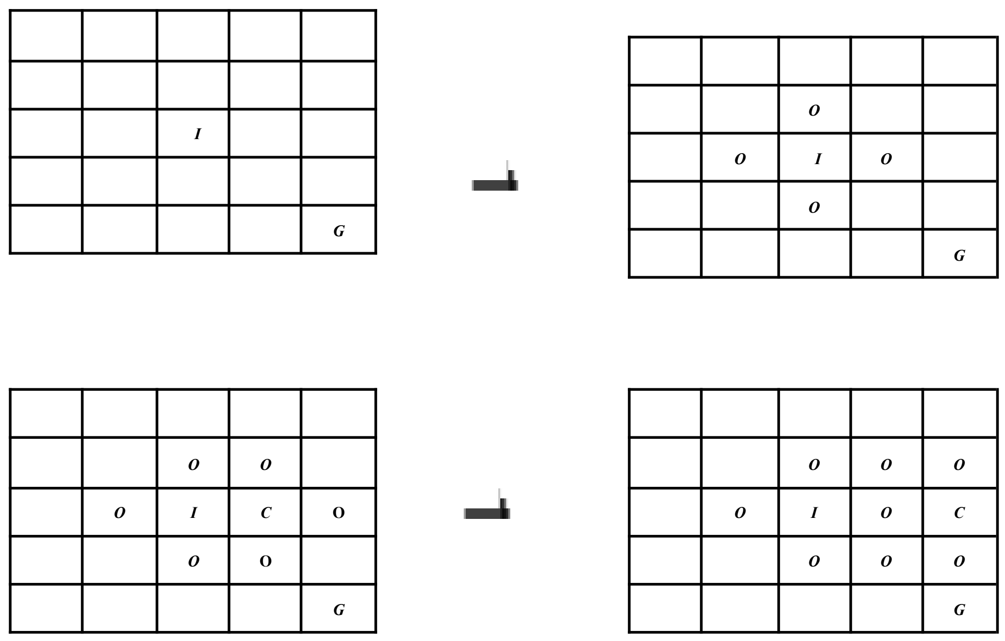
python. Path planning can be implemented from the base information that comes from the given local map. Moreover, cell data from the map is extracted by subscribing to the "*/map*" topic . This extends the usage of *OccupancyGrid* from *nav_msgs.msg*. The data represents a 2D grid map in which each cell represents the probability of occupancy[5] . It uses a tri-state rule with values as 0, 100 and -1. Where '0' represents a free-space cell in the grid. '100' represents a Block cell while '-1' represents an unknown cell (scope is out of map).
On receiving the *OccupancyGrid* we reshape it into a 2D array. The A * path planner takes initial_pair, goal_pair as the parameters. Initially, it finds the neighbouring cells for a given initial point on the map. These cells are appended to an *open_nodes* list.

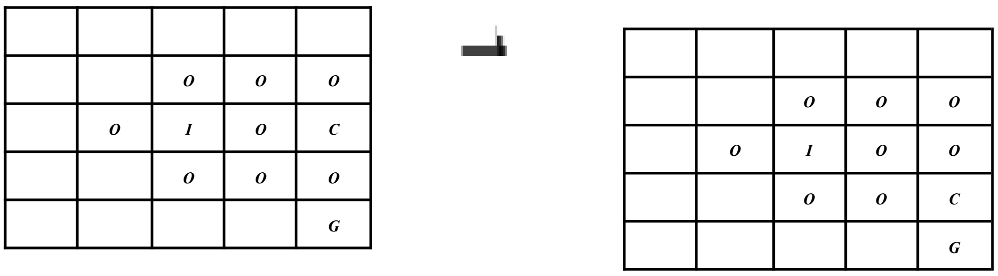
Cost Function in choosing the optimal cells as it explores path to goal state is calculated by *f cost - f(n) = g(n) + h(n).*

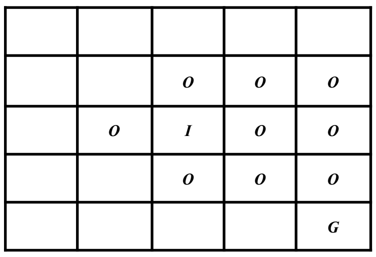
Where *f(n) is the total cost to the goal node.*
g(n) is the distance between the initial node and the current node.
h(n) is the lowest distance from the current to the goal node.

h(n) is calculated by the sum of differences of *x* and *y* coordinates in current node and goal nodes.
For each iteration we set the next current node from the *open_node* list as the lowest cost node. Add this node to the *closed_node* list. Calculates the neighbouring nodes for the current cell and appends these nodes to the list to *open_nodes.*
The above process is iteratively computed for all lowest total cost nodes in the *open_node* list.

Example 1:







The logic of the A star is represented in the below pseudo-code:

```
Astar(initial_position, goal_position):
    Converts the OccupancyGrid data into a 2D array from read_grid().
    Call the find_neighbours() for the initial pair.
    While goal_node not in open_nodes:
        current_pair is set to lowest total cost node by calling the find_lowest_cost_node().
        Removes the current pair from the open nodes and appends it to closed nodes.
        finds the neighbouring nodes for the current_pair by calling find_neighbours()
        Appends these neighbouring nodes to the open nodes.
    Calls the retrace_path() function to retrace the path using the closed nodes list.
    Converts the path into directions by calling convert_path() function.
```

The time and space complexity in implementation of A star is robust and optimal than other search algorithms. Therefore, we developed the path planning section using A star. Time complexity is $O(b^d)$. Where 'd' is the depth i.e. the shortest path to the solution and 'b' is the branching factor representing the average number of successive steps per state. Space complexity in terms of memory A star algorithm is optimal with $O(b^d)$ complexity.

2. Robot Movement

Generation of the possible paths and directions from the A star path planning are the foundation in robot motion in a bi-directional sequence. Robot has the capability of moving in either of the four directions: *top, right, bottom and left*. Each determined by the path in the algorithm for movement of the robot. The readily available rostopic - *geometry_msgs.msg* allows us to access the function *Twist* responsible for ros simulated robot to explore the occupancy grid from the map loaded (here in case we used a .world file of lower ground floor in School of computer science).
Twist() functions embodied in base_data gives scope to move the robot in linear and angular directions. Essentially, *base_data.linear.x* for moving forward w.r.t robot whereas, *base_data.angular.z* helps in robot's rotation in a desired angle.
Linear velocity is set as *0.05 units per sec.*
However, limitations in robot rotation are 0.314 radians per sec. Using a loop for the robot to achieve the angle solves the rotation problem.
Parameters: Directions as a list from A star and angle are passed as arguments to move the robot.
The notion of the robot movement is can be explained in pseudo code below:

```
map_mover(directions, angle):
    Retrieves the directions as a list from A Star
    For each direction:
        Checks the current angle of robot and the direction
        If robot's angle and direction are identical:
            Calls move_robot() and moves forward w.r.t robot
        Else if robot's angle and direction are not identical:
            First calls rotate_robot() sets the desired angle and rotates
            And calls move_robot() and moves  w.r.t robot
```

Scenario 1:
If the robot's angle is 0 degrees w.r.t map frame and the direction is towards 'right'. Then the robot simply moves forward w.r.t robot frame.
Scenario 2:
If the robot's angle is -90 degrees w.r.t map frame and the direction is towards 'left'. Then the robot rotates 90 degrees clockwise and forward w.r.t robot frame.

3. Human Robot Interaction

The components of this framework are *speech to text* and *text to speech*.

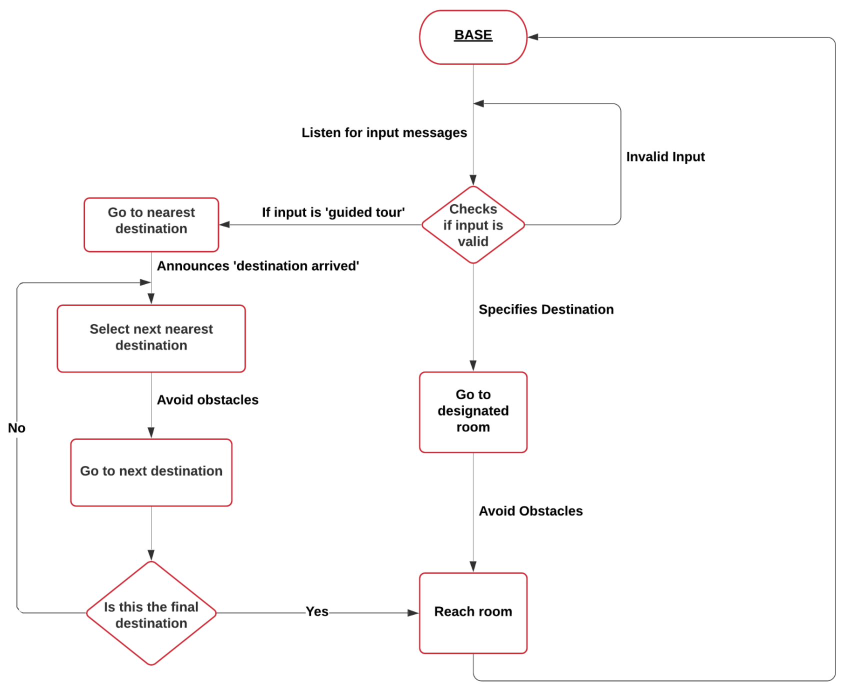
1. Speech to text

This component we use is to personalise the robot and add an interactive mechanism to the robot. Importantly, it recognises the audio from the input, analyses the speech, and converts that to the output, i.e., a text transcript. For developing this segment, we used the Google cloud speech API service. This service API we used as a library is available in python and installed through pip. We use this feature in considering user input and transcribing it to guide and help with the queries of the end-user. This interactive feature enables our project in delivering a sustainable service.

2. Text to speech

As a speech to text service API receives the input, the robot must parse them and analyse the commands received and respond to them with answers. To configure the output, we use python's readily available pytts (python text-to-speech API) library. This library overlaps and maps the input and applies logic for the response to be said by the robot interaction.

The framework combines these *two components* to process user input and respond. When a user requests the robot to navigate them to a destination, this input is first validated. That is, the speech to text system initially validates whether the input is legible and can be converted into text. If it isn't, the system asks the user to repeat.


BASE
Listen for input messages
Invalid Input
Checks if input is valid
If input is 'guided tour'
Go to nearest destination
Announces 'destination arrived'
Specifies Destination
Select next nearest destination
Avoid obstacles
No
Go to designated room
Go to next destination
Avoid Obstacles
Is this the final destination
Yes
Reach room

Afterwards the input text, for eg. "*Can you take me to the Plant room*", the system cleans this text and checks if it's in the controlled input space. If it is, the robot returns the coordinates of the room to the navigation component otherwise the user would be asked to repeat again. The *speech to text* system uses Levenshtein distance to determine how similar the cleaned input string is with the existing rooms.
The flow chart given above gives an overview and explains the process of the whole framework.

Experiments and evaluations

Since we didn't have access to a physical robot, all experiments and results were done using a simulator. As a result we're only able to test the software components of the robot. We're using the *School of Computer Science Pioneer* robot in our simulations.

Components

1. Navigation

Path planning and navigation. This experiment is done in order to validate whether the robot can navigate from its current position on the map to another point on the map.

Experiment

a. The system would choose a random initial state and goal state on the map.
b. The *path generator* generates a path and returns it.
c. Using this path the robot would move across the map.
d. Once the robot reaches the final destination it's final state is compared with the goal state.
e. The test case would pass if the robot is at the goal state otherwise it would fail.

2. Human robot interaction

This experiment is done to determine whether the system is capable of processing the inputs from the user and is able to provide valid responses. This system can handle controlled inputs and works with any voice.

The components being tested are

1. Speech to text
2. Text to speech

These tests are implemented using the python unit testing library.

Experiments

1. Speech to text
   1. The system will be provided with input from the user.
   2. It will then return the predicted output
   3. In case the user asks the robot to take them to a specific room, this output will be a list containing a message, and destination address.
   4. This output is compared against the expected output.
   5. If both are the same then the test case passed otherwise it would fail.

2. Text to speech
    1. The system will receive input from the testing program.
    2. It will then return an output.
    3. In case the input is “Take me to the Grace Hopper lab”, the output should be “Okay let’s head to Grace Hopper lab.”.
    4. This received output is compared against the expected output.
    5. If both are the same then the test case passed otherwise it would fail.

Result

Using the controlled inputs we’ve got the following results. The system currently is capable of handling the input statements given in the table below and the input statements are tested with the 12 points on the map, *medical imaging lab, plant room, teaching lab, robotics lab, lower ground 21, lower ground 23, lower ground 4, lower ground 30b, lower ground 3b, lower ground 3a, lower ground 4 and Mohan’s room.* In total we’ve tested 108 cases, i.e. 9 input statements and 12 points.

Statement	Did all test cases pass	Did test cases fail
Can you take me	✓	
Please take me to	✓	
Where is	✓	
Guide me to	✓	
Show me the way to	✓	
How do i get to	✓	
Can you guide me to	✓	
Can you take me to	✓	
Please show me the path to	✓	

Features

Our current implementation enables a user to either navigate to a specific room on the map or get a tour of a map.

Navigating to a specific room

A user can use the robot to get to a point on the map using this feature. This feature works in the following way,

- While the robot is active, it listens for input from a user.
- Once an input is received it would get validated. This is handled by the human robot interaction framework.
- Once a valid input is received, the robot would process the input and generate a path to the goal state i.e. the room where the user wants to go to. This is handled by the autonomous navigation framework.
- The robot would then follow the path generated and once it reaches the goal state it returns back to its initial position on the map.

Guided tour of a map

A user can use the robot to get to a tour of the map using this feature. This feature works in the following way,

- While the robot is active, it listens for input from a user.
- Once an input is received it would get validated. This is handled by the human robot interaction framework.
- Once the robot receives a valid input, the robot would then generate a path to a room on the map from its initial position. This is handled by the autonomous navigation framework.
- After the robot reaches the goal state or the first room the robot would alert the user and move onto the next room if not already visited.
- This loop runs until all rooms are visited and finally it will return to its homebase i.e. original starting position.

Current limitations and improvement plans

Communication support

1. Instead of just asking where the user wants to head, along with this we could add some minor features and make conversation more engaging.
2. Robot could initiate a conversation on the way to the destination.
3. Since the robot is limited to one floor, the robot will still be able to answer questions that don't relate to a position on the robot's floor. For instance, if someone asks "where is the Grace Hopper lab?", the robot would respond by "Grace Hopper lab is not on this floor, it's on the floor above this."

Diagonal movement

1. Our map is divided into many unit cells
2. In our current version of code, the robot is only capable of moving to the right, left, top, bottom.
3. What we plan on doing is making the robot capable of moving to the top right, top left, bottom left and bottom right.
4. This will open up many more shorter paths to the goal state that make the robot more efficient.

Optimal Path

1. In the current version of code, the robot chooses the path from the direction which has the lowest cost function.
2. With the data from the laser scan we average the range of lowest cost nodes.
3. It chooses the lowest average of nodes in rapid pathfinding from that node to the destination node.
4. This eliminates the cases with trivial nodes/paths and improves algorithm efficiency.

Risk analysis

In this section we look at the potential problems and develop a strategy to reduce its impact.

RISK	DESCRIPTION	LIKELIHOOD	SEVERITY	MITIGATION STRATEGY
Unable to implement a feature	Risk associated to not being able to implement a significant aspect of our autonomous robot	3	5	Attend all lectures, keep up with content taught in lecture and lecture slides. If there is a doubt, ask the lecturer immediately
Poor time management	Lack of time management will make it difficult to keep with deadlines and quality of work	3	5	Raise an issue in the kanban board and assign it during scrum meetings.
Unforeseen illness	N/A		5	Let the team members and module lead know about it discuss what is the best solution
Disagreement between team members	Risk associated to disagreement between team members or any kind of issue between team members	3	5	Discuss between everyone in group about the issue and if there is disagreement on implementation, agree on what the majority says

Conclusion

Robotics is a vast field that should play an active role instead of a passive role and bring about a transformative change in the lives of human beings. In the time of pandemic, to avoid physical contact, one such attempt has been attempted by us in this project. In this paper, we have proposed an idea of an autonomous robot that would act as a guide. It is capable of path finding using A* algorithm, speech recognition implemented using Google Cloud Speech API and text to speech. To confirm the effectiveness of the autonomous robot, we have examined it through many tests which goes into the efficiency of each of the embedded features designed. Experimental results indicate that our system is capable of physically guiding the person with a safe and efficient path in any given space.

Future Work

The problem we have discussed in this document is exciting and challenging for a computer scientist. In the following subsections, we have included our thoughts on what can be done in the coming month and year.

One-month plan

In the coming month after finishing it on a simulator, we would love to try it on an actual physical robot. Along with this, it would be interesting to add some additional minor changes to the physical robot in our project that would help make the robot more efficient. Some of which we could not do because we were working on a simulator.

Long term plan

Future research should adopt innovative research designs to develop and evaluate multi-component autonomous robot interventions for the wellbeing of the world.

Assistance for the visually imparied

One major development would be to use the robot to help visually impaired people navigate unknown environments. Our proposed solution is to use computer vision to identify the visually impaired and the robot would then approach them and ask them if they need any assistance. In case they say yes, the robot will ask them for their destination and navigate them through the building while avoiding obstacles to their destination. The robot would have a bar or a handle for them to hold onto while the robot navigates them to their destination.

Our current hypothesis on how this system would work is, a robot would remain stationary at the entrance of a building. While stationary it would capture images of its environment using a camera. These images would then run through a *custom computer vision model* capable of identifying visually imparied using key features like a white cane, dark glasses, or a guide dog. Once this has been detected, the robot would calculate distance to them using *augmented reality* and move to their position.

This hypothesis vastly simplifies the real world and is an attempt to extend the implementation by [1].

References

1. [1] A tour-guide robot: Moving towards interaction with humans - Biel Piero E. Alvarado Vásquez, Fernando Matía - https://www.sciencedirect.com/science/article/abs/pii/S0952197619302908
2. Hybrid-robotics.berkeley.edu. 2021. [online] Available at: <https://hybrid-robotics.berkeley.edu/publications/ICRA2021_GuideDog.pdf> [Accessed 28 November 2021]
3. Hackaday.io. 2021. *Autonomous UV Robot with SLAM*. [online] Available at: <https://hackaday.io/project/180015-autonomous-uv-robot-with-slam> [Accessed 7 December 2021].
4. Core.ac.uk. 2021. *SLAM algorithm applied to robotics assistance for navigation in unknown environments*. [online] Available at: <https://core.ac.uk/download/pdf/81798022.pdf> [Accessed 7 December 2021].
5. An Approach for 2D Visual Occupancy Grid Map Using Monocular Vision. [online] Available at: cesses <https://www.sciencedirect.com/science/article/pii/S1571066111001824?via%3Dihub>[Accessed 7 December 2021].

Github link

https://github.com/Last-Lost/final-project-intelligent-robotics/tree/main/src/socspioneer